ปัญญา เปรมปรีดี

SOFT MAIL

Good Design/Bad Design

แนะนำโปรแกรมที่ใช้สำหรับบันทึกการซื้อขายหุ้น เพื่อเป็นการวิเคราะห์ปัญหาก่อนการออกแบบระบบงานคอมพิวเตอร์

เวลาที่เราเรียนหนังสือนั้นครูก็มักจะสอนแต่เรื่องของการทำดีหรือทำให้ถูกวิธี ในวิชาที่เกี่ยวกับการออกแบบระบบงานคอมพิวเตอร์ก็เหมือนกัน ครูจะสอนแต่หลักการออกแบบระบบที่ดี อย่างเช่น ระบบจะต้องมีประสิทธิภาพ ระบบจะต้องง่ายต่อการเข้าใจ ระบบจะต้องง่ายต่อการใช้งาน แต่มันน่าแปลกใจที่ว่า ระบบเกือบทุกระบบที่เราพัฒนากันขึ้นมาเองนั้นมีข้อเสียอยู่เสมอ ๆ บางระบบก็ล้มเหลวไปเลย แม้แต่โปรแกรมสำเร็จรูปที่ฝรั่งใช้คนนี่มือยอดเยี่ยม เงินเดือนเป็นแสนเหรียญทำ ก็ยังมีบักและมีข้อตำหนิอยู่อีกเยอะแยะ ดังนั้นเราจึงน่าจะมานั่งคิดกันดู ดูว่าทำไมเจ้าข้อบกพร่องเหล่านี้จึงเกิดขึ้นมาได้ สำหรับผมนั้นได้นั่งมองดูผลงานของตัวเองไปหลายชิ้น แล้วก็มาสรุปได้ว่า **เจ้า "Bad Design" นั้นมันเกิดมาจากความพยามยามที่จะทำตามคำแนะนำของ "Good Design" นั่นแหละ แต่เป็นการทำจนเกินกว่าเหตุ** เรื่องนี้เป็นอย่างไรผมจะขอยกตัวอย่างโปรแกรมอันหนึ่งของผมมาให้ท่านดู

โปรแกรมที่จะยกขึ้นเป็นตัวอย่างนี้เป็นโปรแกรมที่ใช้สำหรับบันทึกการซื้อขายหุ้น มันเป็นโปรแกรมเล็ก ๆ ที่ทำขึ้นสำหรับผู้เล่นหุ้นรายย่อย นั่นคือ จะมีหุ้นอยู่ในมือประมาณ 10-20 ตัวเท่านั้น และจะมีการซื้อและขายปีละไม่ถึง 50 ครั้ง แฟ้มข้อมูลจะมีขนาดที่เตรียมไว้รองรับแค่ 500 รายการ ซึ่งก็น่าจะเพียงพอที่จะใช้งานไปถึง 10 ปี โปรแกรมนี้จะสามารถออกรายงานบนจอภาพได้ว่าหุ้นที่ยังเหลืออยู่ในมือขณะนั้นมีอะไรบ้าง แต่ละตัวมีกำไรหรือขาดทุนเท่าไหร่ และรวมแล้วมีกำไรหรือขาดทุนเป็นกี่เปอร์เซ็นต์ นอกจากรายงานนี้แล้วก็จะมีรายงานที่เกี่ยวกับหุ้นที่ซื้อมาและขายไปแล้ว มันจะออกรายงานในรูปร่างหน้าตาคล้าย ๆ กัน แต่พวกที่ขายไปแล้วนี้จะมีทั้งราคาที่ซื้อมาและที่ขายไป มันจึงแสดงกำไรหรือขาดทุนที่แท้จริงได้เลย

และเพื่อให้ท่านเข้าใจได้ดียิ่งขึ้น ผมก็จะขอแสดงรายละเอียดของเรคอร์ดให้ดู ในลิสติงที่ 1 แสดงการกำหนดลักษณะของเรคอร์ด (Record Layout) ตามภาษาปาสคาล

```
TYPE INVESTTYPE=RECORD
     INVESTNO:INTEGER;
     INVESTNAME:STR8;
     QTY:REAL;
     PARVALUE:REAL;
     BUYPRICE:REAL;
     BUYDATE:STR8;
     CURRPRICE:REAL;
     CURRDATE:STR8;
     SELLPRICE:REAL;
     SELLDATE:STR8;
     STATUS:INTEGER;
   END;
```

ลิสติงที่ 1 Record Layout

จากเรคอร์ดเลเอาต์ดังกล่าวนี้เราจะเห็นได้ว่า ผมแบ่งออกเป็น 11 ฟิลด์ ซึ่งได้แก่ Investment Number, Investment Name, Quantity, Par Value, Buying Price, Buying Date, Current Price, Current Date, Selling Price, Selling Date และ Record Status เจ้า Investment Number นั้นหมายถึง หมายเลขที่โปรแกรมจะกำหนดให้โดยอัตโนมัติ เมื่อผู้ใช้อัพเดตข้อมูลการซื้อ มันจะเป็นหมายเลข 001, 002, 003,.... ไปตามลำดับ มันมีไว้เพื่อการบ่งชี้ว่ามันคือข้อมูลชิ้นที่เท่าใด เจ้า Investment Name ก็คือชื่อของหลักทรัพย์ ตัวนี้จะต้องเป็นตัวอักษรยาว 8 ไบต์ นี่เป็นกฎเกณฑ์การตั้งชื่อย่อที่ตลาดหลักทรัพย์กำหนดขึ้น

เข้า Quantity หรือที่เรียกย่อ ๆ ว่า QTY นั้นจะเป็นจำนวนหุ้น เจ้าตัวนี้ต้องใช้ตัวเลขแบบ Real ทั้งนี้เพราะหุ้นบางตัวในตอนนี้ราคาต่ำกว่า 1 บาท การลงทุนแค่ 1-2 แสนบาท ก็สามารถซื้อได้เกิน 32,767 หุ้น เราจึงใช้การเก็บแบบ Integer ไม่ได้ เจ้า Buying Price ก็คือ ราคาที่ซื้อมาต่อหุ้น ตัวเลขนี้จะมีทศนิยมได้ 2 หลัก เราจึงให้เก็บแบบ Real เจ้า Buying Date คือวันที่ที่ซื้อมา เจ้าตัวนี้จะแยกได้เป็น เดือน ปี และวัน ซึ่งเป็นไปตามแนวที่ตลาดหลักทรัพย์ใช้อยู่ คือมันจะมีรูปร่างเป็น MMMYY-DD หรือคือ 3 ตัวแรกจะเป็นชื่อย่อของเดือน ซึ่งได้แก่ JAN, FEB, MAR, APR, ..., DEC 2 ตัวถัดมาที่เขียนว่า YY คือตัวเลขบอกปี ค.ศ. มันเป็น 2 หลักหลังของปีค.ศ. นั่นคือปี 00 จะหมายถึง ปีค.ศ. 1900 ปี 99 จะหมายถึง ปีค.ศ. 1999 ส่วน 2 ตัวสุดท้ายจะหมายถึงวันที่ นั่นคือ JAN 98-31 จะหมายถึงวันที่ 31 เดือนมกราคม ปีค.ศ. 1998 เจ้า Current Price และ Current Date นั้นจะหมายถึงราคาปิดและวันที่ที่เรานำมาอัพเดตครั้งหลังสุด เจ้า 2 ตัวนี้จะมีความหมายเมื่อเป็นหลักทรัพย์ที่เราซื้อมาแล้วยังถือไว้อยู่ และเราจะต้องอัพเดตใหม่ทุกครั้งที่เราอยากจะรู้ว่าหุ้นตัวนี้มีกำไรหรือขาดทุน เจ้า Selling Price และ Selling Date คือราคาที่ขายไป และวันที่ที่ขายไป ช่องทั้ง 2 นี้ปกติจะทิ้งว่างเอาไว้ (เป็น Spaces) เมื่อไหร่ที่เราบันทึกการขายจึงจะนำข้อมูลเข้ามาใส่ ข้อมูลฟิลด์สุดท้ายคือ Record Status เจ้าตัวนี้จะมีค่าได้ 3 อย่างเท่านั้น คือ 0=Empty, 1=Active, 2=History คำว่า Empty หมายถึงว่าเรคอร์ดนี้ยังว่างอยู่ เราไม่ได้เก็บข้อมูลอะไรไว้ คำว่า Acitve หมายถึง "มีข้อมูลอยู่ และเป็นหลักทรัพย์ที่ซื้อมาแล้วยังไม่ได้ขายออกไป" คำว่า History หมายถึง "มีข้อมูลอยู่แต่เป็นหลักทรัพย์ที่ซื้อมาและขายไปแล้ว"

จากการกำหนดว่า Record Status มี 3 ชนิด คือ 0=Empty, 1=Active และ 2=History นั้นท่านก็คงพอจะเดาได้ว่า **ผมจะเก็บข้อมูลเอาไว้โดยไม่มีการทิ้งไปเลย** นี่จะทำให้เราสามารถตรวจย้อนหลังไปได้ว่าเราได้ทำอะไรไปบ้าง ข้อมูลที่ถูกบันทึกเอาไว้ตรงไหนแล้วจะอยู่ตรงนั้นตลอดไป แฟ้มข้อมูลอันนี้ในตอนแรกจะถูกสร้างขึ้นมาให้เป็นชนิด Empty หมด และมีไว้ครบ 500 เรคอร์ด เมื่อมีการซื้อหุ้นเข้ามาก็จะนำมาบันทึกลงในเรคอร์ดแรก และเปลี่ยน Record Status เป็น 1(=Active) ข้อมูลที่ต้อง

ลิสติงที่ 2
ส่วนของ Main Port

```
BEGIN
  CLRSCR;
  GOTOXY(1,1);WRITE('IV1S01:INVESTMENT RECORDS AND REPORTING');
  SPEEDCHECK;
{  CREATEINVESTFILE;GOTO EOJ;}
  GETCURRENTDATE;
  LOADINVESTBUFF;
  LOADSTOCKNAME;

MAINLOOP:
  CLRSCR;
  GOTOXY(1,1);WRITE('IV1S01:INVESTMENT RECORDS AND REPORTING');
  REPEAT
    GOTOXY(3,3);WRITE('OPTIONS=                    ');
    GOTOXY(3,4);WRITE('   P=PRICE UPDATE           ');
    GOTOXY(3,5);WRITE('   R=REPORT ACTIVE          ');
    GOTOXY(3,6);WRITE('   H=REPORT HISTORY         ');
    GOTOXY(3,7);WRITE('   B=BUY(=ADD INVESTMENT)   ');
    GOTOXY(3,8);WRITE('   S=SELL(=SPLIT INVESTMENT)');
    GOTOXY(3,9);WRITE('   M=MAKE STOCK NAME        ');
    GOTOXY(3,10);WRITE('   A=ADD STOCK NAME         ');
    GOTOXY(3,11);WRITE('   F=FILE MAINTENANCE       ');
    GOTOXY(3,12);WRITE('   E=END                    ');
    READ(KBD,CH);
    IF (CH<>'P') AND (CH<>'R') AND (CH<>'H') AND (CH<>'B') AND
       (CH<>'S') AND (CH<>'M') AND (CH<>'A') AND (CH<>'F') AND
       (CH<>'E') THEN
       BEGIN BEEP;BEEP;END;
    UNTIL (CH='P') OR (CH='R') OR (CH='H') OR (CH='B') OR
          (CH='S') OR (CH='M') OR (CH='A') OR (CH='F') OR
          (CH='E');
  IF (CH='E') THEN GOTO EOJ;
  IF (CH='P') THEN UPDATECURRENTPRICE;
  IF (CH='R') THEN PRINTACTIVE;
  IF (CH='H') THEN PRINTHISTORY;
  IF (CH='B') THEN ADDINVESTMENT;
  IF (CH='S') THEN UPDATEINVESTMENT;
  IF (CH='M') THEN CREATESTOCKNAMEFILE;
  IF (CH='A') THEN ADDSTOCKNAME;
  IF (CH='F') THEN FILEMAINTENANCE;
  GOTO MAINLOOP;

EOJ:
{  UNLOADINVESTBUFF;}
  GOTOXY(1,24);WRITE('END OF JOB       ');
END.
```

นำเข้ามาบันทึกไว้ในตอนซื้อก็คือ ชื่อหลักทรัพย์ (Investment Name) จำนวนหุ้น (Quantity) ราคาพาร์ (Par Value) ราคาซื้อ (Buying Price) และวันที่ซื้อ (Buying Date) โปรแกรมจะกำหนดหมายเลขประจำเรคอร์ด (Investment Number) ให้ และจะเอาราคาซื้อกับวันที่ซื้อไปใส่ในราคาปิดล่าสุด (Current Date) ให้โดยอัตโนมัติ และเมื่อมีการขายออกไปเราก็จะมาอัพเดตการขาย ซึ่งจะต้องใส่ข้อมูล 3 ชิ้น คือจำนวนหุ้นที่ขาย (Quantity) ราคาที่ขายไป (Selling Price) และวันที่ที่ขายไป (Selling Date) ในกรณีที่ขายหมดทั้งจำนวน (Quanlity ขาย=Quantity ในเรคอร์ด) โปรแกรมก็จะเอาราคาขายและวันที่ที่ขายไปใส่ในช่องทั้ง 2 แล้วเปลี่ยน Record Status เป็น 2 (=History) แต่ถ้าเราขายไปเพียงบางส่วน (Quantity ขาย < Quantity ในเรคอร์ด) เราจะต้องแยกเอาส่วนที่เหลือไปตั้งเป็นเรคอร์ดใหม่ ซึ่งจะเป็นเรคอร์ดที่ว่างเปล่าอันแรกที่เหลืออยู่ และยังคงค่าของ Record Status เป็น 1 (=Acitve) อยู่ต่อไป

ลอจิกของโปรแกรมนี้ดูง่ายดี ลิสติงที่ 2 แสดงส่วนที่เป็น Main Part ของโปรแกรมนี้ โปรแกรมนี้จะขึ้นข้อความบนหน้าจอภาพให้ท่านเลือก (Options) ท่านกดแป้นตัวอักษรตามที่ต้องการเพียงครั้งเดียวมันก็จะไปทำตามสั่ง ทำเสร็จแล้วก็จะวนกลับมาที่เก่า ทางเลือกอันสุดท้ายคือ E-END ซึ่งจะหมายถึงการจบโปรแกรม โปรแกรมจะกระโดดไปที่ EDJ: และแสดงข้อความว่า "END OF JOB" ที่บรรทัดที่ 24 แล้วจบไป

การออกแบบระบบโดยให้สร้างแฟ้มข้อมูลเอาไว้เต็ม 500 เรคอร์ด แล้วมาอัพเดตให้เป็น Acitve และ History ในภายหลัง แบบนี้เป็นวิธีการที่จะช่วยป้องกันปัญหาได้หลายอย่าง ประการแรกคือ จะไม่มีการขอเพิ่มเนื้อที่ในจานแม่เหล็กอีกเลย มันจึงไม่มีการเกิด "Abnormal End Of Job" เพราะสาเหตุจาก "No Disk Space" ข้อดีประการที่ 2 คือ การอัพเดตแฟ้มข้อมูลจะใช้เวลาสั้น โอกาสที่จะเกิดไฟฟ้าดับในระหว่างการอัพเดตข้อมูลจึงมีน้อยมาก ข้อมูลจึงอยู่ครบถ้วนตลอดเวลา และถ้าท่านขุดลึกลงไปในขั้นตอนของการอัพเดตจริง ๆ แล้วจะพบว่าโปรแกรมจะวิ่งใหม่ได้ทุกครั้ง ไม่ว่าจะโดนใครไปเตะปลั๊กจนไฟดับสักกี่ครั้ง ข้อดีประการที่ 3 คือ ขั้นตอนการทำงานจะเหมือนกับการใช้สมุดบันทึกธรรมดา ๆ ซึ่งจะตรงกับความคิดของคนโดยทั่ว ๆไป มันจึงอธิบายให้ยูสเซอร์ฟังได้ง่าย และถ้าใครจะต้องมาแก้ไขหรือปรับปรุงโปรแกรมก็จะทำได้โดยง่าย ยกตัวอย่างเช่น เมื่อจะออกรายงานหุ้นที่ยังคงเหลืออยู่ในมือ (option= R=Report Active) โปรแกรมก็จะไล่จากเรคอร์ดที่ 1 ไปจนถึงสุดท้าย อันไหนที่มี Record Status เป็น 1 (=Active) ก็จะถูกดึงออกมาแสดงเป็น 1 บรรทัดบนจอภาพ ในขณะที่ไล่ไปตามลำดับนั้นก็จะสะสมยอดกำไร-ขาดทุนและต้นทุนทั้งหมดไปด้วย เมื่อจบทุกเรคอร์ดแล้วก็จะแสดงยอดกำไรหรือขาดทุน และคำนวณเป็นเปอร์เซ็นต์ของต้นทุนได้ การออกรายงานดูประวัติเก่า (Option =H=Report History) ก็เช่นกัน เราจะไล่จากเรคอร์ดที่ 1 ไปจนถึงสุดท้าย อันไหนมี Record Status เป็น 2 (=History) ก็จะถูกดึงออกมาแสดงและเก็บยอดกำไร-ขาดทุนและต้นทุนเอาไว้

มันจะมีอยู่อันเดียวเท่านั้นที่ดูจะยุ่งยากสักหน่อย สำหรับผู้ใช้ คือ ก่อนที่จะออกรายงาน Active นั้นเราควรทำการอัพเดตราคาปิด (Current Price) ให้มันล่าสุดเสียก่อน นี่คืออ็อปชัน P (=Price Update) ถ้าเราใช้ลอจิกแบบไล่จากเรคอร์ดที่ 1 ไปจนถึงอันสุดท้ายที่ Active อยู่ ผู้ใช้ก็จะต้องไปวิ่งหาข้อมูลมาคีย์เข้าไปตามลำดับเหล่านั้น เรื่องนี้จะไม่ยุ่งยากเลยถ้าเราสามารถเข้าถึงข้อมูลของตลาดหลักทรัพย์ แต่การขอเป็นสมาชิกของตลาด และการก๊อบปี้ข้อมูล "ตารางหุ้น" จากตลาดหลักทรัพย์นั้นจะมีค่าใช้จ่ายสูงมาก มันไม่เหมาะกับผู้เล่นหุ้นรายย่อย (อย่างตัวผมเป็นต้น) ค่าสมาชิกนั้นเดือนละ 3-4 หมื่นบาท แล้วยังต้องทำ File Transfer ทุกวัน วันละ 5-10 นาที และในตอนนี้ตลาดหลักทรัพย์ก็ได้เปลี่ยนฟอร์แมตไปมากจนทำให้แฟ้มข้อมูล "ตารางหุ้น" ดังกล่าวมันใหญ่โตมโหฬารจนไม่มีที่เก็บในดิสก์ ทางออกในเรื่องนี้ก็คือ ต้องซื้อหนังสือพิมพ์ที่มีตารางหุ้นรายวันมาใช้ ถ้าเรามีหุ้นอยู่ในมือแค่ 10-20 ตัว เราก็จะต้องเปิดหนังสือพิมพ์แล้วคีย์เข้าไปแค่ 10-20 ตัวเลขเท่านั้น เรื่องนี้แต่เดิมผมก็ใช้ลอจิกแบบง่าย ๆ คือไล่จากเรคอร์ดที่ 1 ไปจนถึงอันสุดท้าย อันไหนมันมี Record Status เป็น 1 (=Active) ก็ให้มันหยุดและเรียกอินพุตจากคีย์บอร์ด

แต่:... เมื่อใช้ไปสักพักหนึ่งก็เกิดความรู้สึกว่าไม่สะดวก กล่าวคือ หุ้นที่เราซื้อมานั้นมันไม่เรียงตามลำดับของตารางหุ้น เรื่องนี้เรายังคับไม่ได้ วันแรกเราอาจพอใจที่จะซื้อหุ้นแบงค์กรุงเทพ (=BBL) วันถัดมาเกิดพอใจไปซื้อหุ้นชินวัตร (SHIN) อีก 2 สัปดาห์ต่อมามีคนมาชวนให้ซื้อหุ้นห้องเย็นเอเซียนซีฟู้ด (ASIAN).... ฯลฯ หุ้นเหล่านี้มันอยู่กันคนละกลุ่ม **ผู้ใช้ก็ต้องพลิกหนังสือพิมพ์ไปมา** เมื่อมีหุ้นอยู่ในมือ 10-20 หุ้นก็ต้องพลิกหาไป 10-20 ครั้ง มันจึงไม่สะดวกอย่างยิ่ง

```
PROCEDURE UPDATECURRENTPRICE;
VAR I,K,FOUND,UPDATESWT:INTEGER;
VAR TARGETSTOCKNAME:STR8;
VAR TARGETSTOCKPRICE:REAL;
BEGIN
  UPDATESWT:=0;
  FOR I:=1 TO LASTSTOCKNAME DO
    BEGIN
    TARGETSTOCKNAME:=STOCKNAME[I];
    FOUND:=0;
    FOR K:=1 TO LASTBUFF DO
      IF (BUFF[K].INVESTNAME=TARGETSTOCKNAME) AND
         (BUFF[K].CURRDATE<>CURRENTDATE) THEN FOUND:=1;
    IF FOUND=1 THEN
      BEGIN
      REPEAT
        GOTOXY(1,22);WRITE('NEW PRICE OF ',TARGETSTOCKNAME,'=');
        GOTOXY(1,23);WRITE('   <0.00..2000.00>                ');
        ACCEPTNUM(23,22,7,2,TARGETSTOCKPRICE);
        IF (TARGETSTOCKPRICE<0.00) OR (TARGETSTOCKPRICE>2000.00) THEN
          BEGIN BEEP;BEEP;END;
        UNTIL (TARGETSTOCKPRICE>=0.00) AND (TARGETSTOCKPRICE<=2000.00);
      FOR K:=1 TO LASTBUFF DO
        IF BUFF[K].INVESTNAME=TARGETSTOCKNAME THEN
          BEGIN
          BUFF[K].CURRPRICE:=TARGETSTOCKPRICE;
          BUFF[K].CURRDATE:=CURRENTDATE;
          END;
      UPDATESWT:=1;
      END;
    END;
  GOTOXY(1,22);WRITE('                                          ');
  GOTOXY(1,23);WRITE('                                          ');
  IF UPDATESWT=1 THEN
    BEGIN
    UNLOADINVESTBUFF;
    LOADINVESTBUFF;
    END;
END;
```

ลิสติงที่ 3
ลอจิกของ Price Update

แล้ว.... เข้าคำว่า "ต้องสะดวกต่อผู้ใช้" ก็เลยทำให้ผมต้องออกแบบเพิ่มเติม วิธีการก็คือ ผมจะให้มีแฟ้มข้อมูลอันที่ 2 ขึ้นมา ในแฟ้มนี้จะเก็บรายชื่อของหลักทรัพย์เรียงกันตามตารางหุ้น หรือคือเรียงตามที่ตลาดหลักทรัพย์เขาจัดเอาไว้ (แยกเป็นกลุ่ม ๆ แต่ละกลุ่มเรียงตามชื่อหลักทรัพย์) แล้วมาปรับลอจิกของโปรแกรมที่ทำเรื่อง Price Update ดังนี้

(1) ไล่ตามลำดับในชื่อของแฟ้มข้อมูลที่ 2

(2) จากชื่อที่หยิบมานั้น ให้ไปเช็คดูในแฟ้มข้อมูลที่ 1 จากเรคอร์ดที่ 1 ไปจนถึงสุดท้าย ถ้ามีหลักทรัพย์ชื่อนี้และยัง Active อยู่ (Status=1) ก็ให้หยุดรับข้อมูลจากคีย์บอร์ด

(3) ทุกครั้งที่ได้ข้อมูลจากคีย์บอร์ดก็ให้ไล่อัพเดตค่าของราคาปิด (Current Price) และวันที่ (Current Date) ในแฟ้มข้อมูลแรกที่มีชื่อเดียวกันทั้งหมด

รายละเอียดของลอจิกนี้ มีแสดงอยู่ในลิสติงที่ 3

จากลิสติงที่ 3 นั้นเราจะเห็นได้ว่าเขียนได้ไม่ยากเลย และจะช่วยให้ไม่ต้องพลิกหนังสือพิมพ์กลับไปกลับมา แต่มันก็จะมีคำถามว่าแฟ้มข้อมูลที่ 2 เก็บรายชื่อของหลักทรัพย์เรียงตามแบบที่กำหนดโดยตลาดหลักทรัพย์นั้นจะมาจากไหน และถ้ามีการเพิ่มเติมหรือไล่ออกจากตลาดหลักทรัพย์เราจะทำอย่างไร

เรื่องนี้ทำให้ผมต้องเพิ่มโพรซีเดอร์และออปชันขึ้นมาอีก 2 คู่ คือ M=Make Stock Name และ A=Add Stock Name แนวคิดในเรื่องนี้ก็คือ จริง ๆ แล้วเราไม่จำเป็นต้องมีรายชื่อของหลักทรัพย์ครบตามที่มีอยู่ในตลาดหลักทรัพย์ เราเก็บแต่เฉพาะที่เราต้องการใช้ในระบบของเราเท่านั้น นั่นคือเก็บแต่หลักทรัพย์ที่เราเคยซื้อมา และชื่อเหล่านี้ไม่จำเป็นต้องลบทิ้งเมื่อหลักทรัพย์อันนั้นถูกไล่ออก ยิ่งไปกว่านั้น การเปลี่ยนชื่อก็ไม่มีด้วย การอัพเดตจึงมีอยู่อย่างเดียว คือการ "insert" ชื่อใหม่เข้ามา เราไม่ต้องมี "add", "change" หรือ "delete" ในการที่จะ "insert" นี้ เราจะใช้วิธีเอาชื่อมาแสดงบนจอภาพ 10 ชื่อ ถ้าผู้ใช้ต้องการ "insert" ในช่วงนี้ก็กดแป้นตัว I ถ้าไม่ต้องการก็ให้กดแป้น F เพื่อไปข้างหน้า (Forward) หรือกดแป้น B เพื่อถอยหลัง หรือกดแป้น E เพื่อออกไปจากโพรซีเดอร์นี้ การข้ามไปข้างหน้าหรือถอยหลังนี้จะทำทีละ 10 ชื่อ ด้วยวิธีนี้เราจะหาจุดที่จะ "insert" ได้อย่างรวดเร็ว รายละเอียดของโปรแกรมส่วนนี้มีแสดงอยู่ในลิสติงที่ 4

ลิสติงที่ 4
ลอจิกของการ
Add Stock Name

```
PROCEDURE ADDSTOCKNAME;
VAR J,K,ADDSWT,DISPLAYFROM,LN,N,INSERTAT,ERRSWT:INTEGER;
VAR NAME:STR8;
VAR CH:CHAR;
BEGIN
  CLRSCR;
  GOTOXY(1,1);WRITE('IV1S01:INVESTMENT RECORDS AND REPORTING');
  GOTOXY(1,2);WRITE('          <ADD STOCK NAME>              ');
  ADDSWT:=0;
  DISPLAYFROM:=1;
  REPEAT
    DISPLAYSTOCKNAME(DISPLAYFROM);
    REPEAT
      GOTOXY(1,22);WRITE(' F=FORE, B=BACK, I=INSERT');
      GOTOXY(1,23);WRITE(' E=EXIT                    ');
      READ(KBD,CH);
      IF (CH<>'F') AND (CH<>'B') AND (CH<>'I') AND (CH<>'E') THEN
        BEGIN BEEP;BEEP;END;
      UNTIL (CH='F') OR (CH='B') OR (CH='I') OR (CH='E');
    CLEARLINE(22);CLEARLINE(23);
    IF (CH='F') THEN
      BEGIN
      IF (DISPLAYFROM+10<=LASTSTOCKNAME) THEN
        DISPLAYFROM:=DISPLAYFROM+10 ELSE
        BEGIN BEEP;BEEP;END;
      END;
    IF (CH='B') THEN
      BEGIN
      IF (DISPLAYFROM>1) THEN
        DISPLAYFROM:=DISPLAYFROM-10 ELSE
        BEGIN BEEP;BEEP;END;
      END;
    IF (CH='I') THEN
      BEGIN
      REPEAT
        GOTOXY(1,22);WRITE(' INSERT AT NO ?');
        ACCEPTINTEGER(17,22,4,INSERTAT);
        ERRSWT:=0;
        IF (INSERTAT<DISPLAYFROM) OR (INSERTAT>DISPLAYFROM+10-1) THEN
          BEGIN BEEP;BEEP;ERRSWT:=1;END;
        IF (INSERTAT>LASTSTOCKNAME+1) THEN
          BEGIN BEEP;BEEP;ERRSWT:=1;END;
        UNTIL ERRSWT=0;
      REPEAT
        GOTOXY(1,23);WRITE(' STOCK NAME = ? <STRING[8]>          ');
        ACCEPTSTRING(29,23,8,NAME);
        ERRSWT:=0;
        FOR K:=1 TO LASTSTOCKNAME DO
          IF NAME=STOCKNAME[K] THEN
            BEGIN BEEP;BEEP;ERRSWT:=1;END;
        UNTIL ERRSWT=0;
      IF NAME<>'        ' THEN
        BEGIN
        FOR J:=LASTSTOCKNAME DOWNTO INSERTAT DO
          STOCKNAME[J+1]:=STOCKNAME[J];
        STOCKNAME[INSERTAT]:=NAME;
        LASTSTOCKNAME:=LASTSTOCKNAME+1;
        ADDSWT:=1;
        END;
      CLEARLINE(22);CLEARLINE(23);
      END;
    UNTIL (CH='E');
  IF ADDSWT=1 THEN
    BEGIN
    UNLOADSTOCKNAME;
    LOADSTOCKNAME;
    END;
END;
```

หลังจากปรับแก้โปรแกรมและเพิ่มแฟ้มข้อมูลที่ 2 เข้าไปแล้ว ได้เกิดเหตุการณ์อะไรบ้าง?

ถ้ามองทางด้านผู้ใช้ผมก็พบว่ามันสะดวกขึ้น แต่ไม่ช้าไม่นานมันก็เกิดปัญหา ประการแรกก็คือ มีการ "insert" ชื่อหลักทรัพย์เข้าไปผิดที่ได้ การควบคุมเรื่องนี้ไม่มีอยู่ในโปรแกรม ผู้ใช้ต้องใช้ความระมัดระวังด้วยตนเอง และเนื่องจากเราไม่มีออปชันอะไรที่จะไปลบ (delete) ชื่อที่ผิดนี้ได้ มันก็จะคงอยู่ผิดที่อย่างนี้ไปตลอดชีวิต ซึ่งทำให้ต้องพลิกหน้าหนังสือพิมพ์ทุกครั้งที่อัพเดตราคาปิด (Current Price) ของหลักทรัพย์ตัวนี้ หนทางแก้จริง ๆ นั้นก็มีอยู่ คือต้องใช้ออปชัน M=Make Stock Name แล้วคีย์ชื่อหลักทรัพย์ใหม่หมด

ข้อผิดพลาดประการที่ 2 คือ ในตอนแรกผมไม่ได้เช็คการมีชื่อซ้ำ ซึ่งมีผลทำให้เกิดการ "insert" ชื่อที่ซ้ำกันเข้าไปได้ แล้วก็ต้องไปแก้ปัญหาด้วยการมาเพิ่มเติมลอจิกเพื่อป้องกันการเติมชื่อซ้ำ แล้วยังต้องไปเพิ่มลอจิกในตอนอัพเดต Current Price ให้มันตรวจเสียก่อนว่า Current Date มันเท่ากับ "วันนี้" หรือไม่ ถ้าเท่ากันก็แสดงว่ามีการอัพเดตมาแล้ว จึงไม่จำเป็นต้องทำซ้ำ แต่อย่างไรก็ตาม เมื่อเติมเงื่อนไขอันนี้เข้าไปแล้วก็จะทำให้อัพเดตได้วันละครั้งเดียว ถ้าเกิดคีย์ค่าของ Current Price เข้าไปผิดและอัพเดตไปแล้ว เราก็จะต้องรอถึงพรุ่งนี้จึงจะแก้ไขใหม่ได้

ถ้ามองทางด้านผู้พัฒนาและบำรุงรักษา ผมก็พบว่าการเติมแฟ้มข้อมูลที่ 2 เข้ามานี้สร้างความวุ่นวายและสับสนขึ้นอีกแยะ และมีโอกาสที่ทิ้งบักเอาไว้อีกหลายที่ ความไม่เรียบร้อยของโปรแกรมตรงส่วนนี้ทำให้ผมคิดฟุ้งซ่านไปอีกหลายอย่าง เช่น เมื่อวิเคราะห์ซ้ำใหม่แล้วก็จะเห็นได้ว่า ชื่อกลุ่มของหลักทรัพย์และชื่อของหลักทรัพย์นั้นเป็น sort key ของเรื่องนี้ ดังนั้นถ้าเราเก็บชื่อกลุ่มเข้าไว้ด้วยกันก็จะไม่ต้องมีแฟ้มข้อมูลที่ 2 แต่การที่จะเพิ่มฟิลด์ใหม่ (ชื่อของกลุ่มหลักทรัพย์) เข้ามานี้จะทำให้ต้องปรับแก้โปรแกรมอีกมากมาย มันเท่ากับการรื้อโครงสร้างใหม่หมด มันจึงทำไม่ได้ หรือไม่น่าทำ... ฯลฯ

เมื่อมองย้อนไปย้อนมาแล้วผมก็สรุปได้ว่า Good Design ของผมก็ได้กลายไปเป็น Bad Design โดยปริยาย จุดเริ่มต้นของความวุ่นวายนั้นมาจากความพยายามที่จะปรับปรุงให้ดีขึ้น แต่ลงท้ายมันกลายเป็นให้ผลร้ายกับระบบงาน

เห็นไหมครับ Bad Design มันเกิดจากความพยายามที่จะออกแบบให้ดีจนเกินกว่าเหตุ เรื่องอย่างนี้จะมีอีกมากมายในระบบงานต่าง ๆ ที่เรามีอยู่ แม้แต่ปัญหาเรื่อง Y2K (Year 2000) นั้นก็เป็นเรื่องเดียวกัน ความพยายามที่จะทำให้ระบบงานมีประสิทธิภาพมากที่สุด หรือประหยัดให้ได้มากที่สุด เราก็เลยตัดเอา 2 ดิจิตแรกของปีค.ศ. ออก ตัดลงให้เหลือแค่ 2 ดิจิตหลัง เราสามารถประหยัดเนื้อที่ในจานแม่เหล็กไปได้เยอะแยะ แต่พอใกล้ถึงปีค.ศ. 2000 เราก็มาบ่นกันว่าจะเอาเงินและแรงงานที่ไหนมาแก้ไขโปรแกรมและข้อมูลเพื่อให้ปี "00" นั้นมันหมายถึงปีค.ศ. 2000 ไม่ใช่ปีค.ศ. 1900

ข้อสรุป ข้อแนะนำของผมในเรื่องนี้ก็คือ เวลาจะออกแบบอะไรก็ตาม ขอให้ลิสต์ข้อดีและข้อเสียออกมาให้หมด แล้ววิเคราะห์ข้อเสียเหล่านั้นอย่างจริงจัง **ข้อเสียที่เราเห็นว่าเล็ก (เพราะอยากจะประหยัดหรือเพิ่มประสิทธิภาพ หรือเพิ่มความสะดวกในการใช้งาน) นั้น มันมักจะกลายเป็น "ใหญ่" ในภายหลัง**

โปรแกรมตัวอย่างนี้ ถ้าท่านต้องการจะนำไปศึกษาหรือใช้งานก็ขอให้ติดต่อบริการซอฟต์เมล์ โปรแกรมนี้ชื่อว่า IV1S01.COM มันมีซอร์สโค้ดชื่อ IVIS01.PAS มีแบคอัพชื่อ IV1S01.BAK ภาษาที่ใช้เขียนคือ Turbo Poseal Version 3.0 ของ Borland International มันวิ่งภายใต้ดอส 3.1 ขึ้นไป และใช้กับพีซีรุ่นไหนก็ได้ มันไม่ต้องการความเร็วของซีพียู ตัวแฟ้มข้อมูลชื่อ IV1D01.DAT และ IV1DO2.DAT ตามลำดับ มันกินเนื้อที่ในจานแม่เหล็กนิดเดียว (ประมาณ 50 กิโลไบต์) และเมื่อรวมทั้งระบบแล้วกินเนื้อที่ไม่ถึง 200 กิโลไบต์ นักเลงหุ่นน่าจะสนใจ

ขอให้โชคดีกับการออกแบบระบบงานคอมพิวเตอร์ในอนาคตต่อ ๆ ไปนะครับ